



# IJEAST

INTERNATIONAL JOURNAL  
OF ENGINEERING APPLIED SCIENCE  
AND TECHNOLOGY



**VOLUME : 11    ISSUE : 01    Print / Issue Publication Date: 02-Jun-2026**



**ISSN : 2455-2143**



**DOI : 10.33564/IJEAST.2026.v11i01.021**

Indexed In



[WWW.IJEAST.COM](http://WWW.IJEAST.COM)

[editor@ijeast.com](mailto:editor@ijeast.com)

# GREEDY GRID SCHEDULING ALGORITHM IN STATIC ENVIRONMENT

Saumitra Singh  
Pratapgarh, Uttar Pradesh, India

**Abstract**— Grid scheduling is a technique by which the user demands are met and the resources are efficiently utilized. The scheduling algorithms are used to minimize the jobs waiting time and completion time. Most of the minimization algorithms are implemented in homogeneous resource environment. In this paper the presented algorithm minimize average turnaround time in heterogeneous resource environment. This algorithm is based on greedy approach which is used in static job submission environment where all the jobs are submitted at same time. Taken all jobs independent the turnaround time of each job is minimized to minimize the average turnaround time of all submitted jobs.

**Keywords** - greedy; grid; heterogeneous; high performance computing; scheduling.

## I. INTRODUCTION

Using the distributed resources to solve the applications involving large volume of data is known as grid computing [1], [2]. There exist many tools to submit jobs on the resources which have different computational power and are connected via Local Area Network (LAN) or Virtual Private Network (VPN). The main challenge in grid computing is efficient resource utilization and minimization of turnaround time. The existing system model consists of the web based grid network platform with different management policies, forming a heterogeneous system where the computing cost and computing performance become significant at each node [3], [4].

In grid computing environment, applications are submitted for use of grid resources by users from their terminals. The resources include computing power, communication power and storage. An application consists of number of jobs; users want to execute these jobs in an efficient manner [5]. There are two possibilities of submission of jobs/data on resources; in one of them, job is submitted on the resources where the input data is available and in the other, on the basis of specific criteria, resource is selected on which both job and input data are transferred. This paper uses second approach, wherein the job is submitted on a scheduler and data on a resource identified by the scheduler. A resource in existing algorithms is selected randomly, sequentially or according to its processing power [2], [6], [7]. In this paper the proposed algorithm chooses a resource on the basis of

processing power, job requirement and time to start at that resource.

The next section describes details about system model. Section 3 describes the proposed scheduling algorithm in static job submission environment. In section 4 the experimental details and the results of experiments are presented with comparison of some existing algorithms. In section 5 conclusions and suggestions for future improvements are proposed.

## II. SYSTEM MODEL

A grid is considered as the combination of multiple layers. In our model the whole system is composed of three layers (Fig.1). The first layer is the user application layer in which the user authentication is done and jobs are submitted to the scheduler by the user. The second layer contains scheduler and GIS. The scheduler schedules jobs among various resources after taking resource status information from GIS. The second layer is connected through a VPN to user. VPN provides additional security and only authorized users can access services. All the resources reside in third layer where user's jobs are executed which are also connected through VPN.

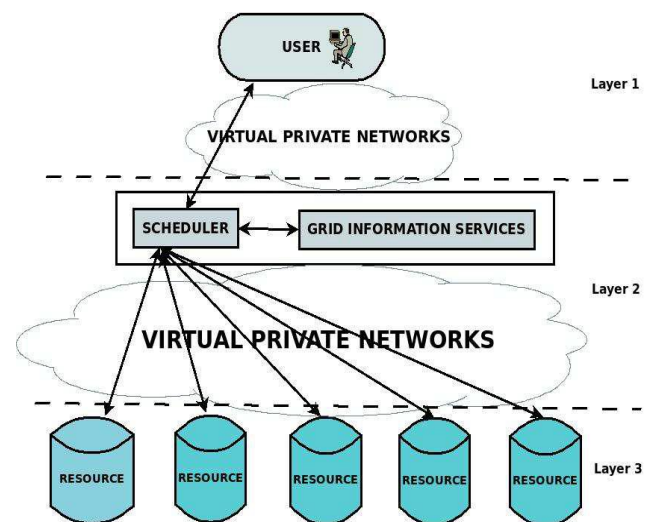


Figure 1. Layered architecture

## III. PROPOSED ALGORITHM

The existing grid scheduling algorithms are based on the speed of resources [6], [7]. Each resource of layer 3



(Figure. 1) has different processing power and all the resources of layer 3 are connected via homogeneous communication environment in which the communication delay between scheduler and resources is assumed constant, also the jobs are assumed to be submitted on layer 1 having different job requirements.

An algorithm is proposed in this paper which is suitable for static job submission in a heterogeneous resource environment connected to the scheduler through a homogeneous communication environment. A greedy approach is used to solve the job scheduling problem. According to the greedy approach "A greedy algorithm always makes the choice that looks best at that moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution" [8]. The proposed algorithm uses a similar approach; it takes every job as independent of each other and each of them is scheduled on a resource to give minimum turnaround time for that job. The overall turnaround time of all the jobs is thus minimized. The parameters used in this algorithm are as follows:

A set of resources,  $R = \{R_1, R_2, R_3, \dots, R_n\}$ .

$J_i$  = The submitted  $i^{\text{th}}$  job.

$Arr\_time_i$  = Arrival time of job  $J_i$ .

$Proc\_power_j$  = Processing power of resource  $R_j$ .

$Strt\_time_j$  = Estimated time at which a job starts execution at resource  $R_j$ .

$Job\_req_i$  = Length of job  $J_i$ .

$Schd\_value_{ij}$  = Expected turnaround time of  $i^{\text{th}}$  job at  $j^{\text{th}}$  resource.

$Min$  = The minimum of  $Schd\_value_{ij}$  among all resources.

$Res\_id$  = Current selected resource id having optimum turnaround time.

The algorithm used to schedule a job is given as follows:

#### **GREEDY\_SCHEDULE**

```

/*The users submit their jobs on the scheduler.*/ For all
resource  $R_j$ 
/*Initialize the start time at resources.*/  $Strt\_time_j=0.0$ 
End For
/*The jobs are stored in a queue Q.*/
Insert all the jobs  $J_i$  in Q
While Q is not empty
do
Delete the job  $J_i$ 
SUBMIT_NEW_JOB
UPDATE_STATUS
Advance the Q pointer
End While

```

End GREEDY\_SCHEDULE

The scheduler uses SUBMIT\_NEW\_JOB algorithm to find the best suited resource that minimizes the turnaround time.

The turnaround time is calculated on the basis of expected completion time of a job.

The detailed SUBMIT\_NEW\_JOB algorithm is as follows:

#### **SUBMIT\_JOB**

```

Min =  $\infty$ 
For every resource  $R_j$ 
/* Calculating the expected turnaround time*/
 $Schd\_value_{ij} = Strt\_time_j + (Job\_req_i/Proc\_power_j)$ 
If Min is greater than  $Schd\_value_{ij}$ 
Then
Min =  $Schd\_value_{ij}$ 
Res_id =  $R_j$ 
End If
End For
Submit the job  $J_i$  to Res_id resource
Submit the input data of  $J_i$  job to Res_id resource
End SUBMIT_NEW_JOB

```

Once the scheduler submits a job to a resource, the resource will remain for some time in processing of that job. The UPDATE\_STATUS algorithm is used to find out when the resource will be available to process a new job. The UPDATE\_STATUS algorithm is given below:

#### **UPDATE\_STATUS**

```

/* Res_id is the resource on which the job  $J_i$  is
submitted. j is the index of resource on which the job  $J_i$ 
is submitted and  $R_j = Res\_id$ */
 $Strt\_time_j = Schd\_value_{ij}$ 
End UPDATE_STATUS

```

The above presented algorithm has the time complexity of  $O(n)$  for each job, where  $n$  is the number of resources. The above algorithm required additional space to store the resources current status for availability.

## **IV. EXPERIMENTAL RESULTS**

The GridSim simulator [6] is used to simulate the algorithms. The GridSim toolkit is used to simulate heterogeneous resource environment and the communication environment. The experiments are performed with three algorithms. The algorithms are Random Resource Selection and Equal Job Distribution and Proposed Algorithm. The input data is taken to be the same for all the three algorithms. The simulation is conducted with three resources which are shown in Table I.



TABLE I. RESOURCES WITH THEIR ARCHITECTURE AND PROCESSING POWER.

Resource	R0	R1	R2
Architecture	Sun Ultra	Sun Ultra	Sun Ultra
OS	Unix	AIX	Unix
Proc_power(in MIPS)	48000	43000	54000

The scheduler submits these jobs on resources according to these algorithms. The algorithms are presented one by one with their simulation results.

**A. Random Resource Selection**

In this algorithm the scheduler contacts GIS to obtain the resource information and then it chooses a resource randomly [7]. The job is submitted on this chosen resource. This algorithm is very simple to implement and has less overhead on the scheduler. The bar chart (Fig. 2) shows the turnaround time of different jobs. The completion time is a time at which the result of a job is available. After simulation the average turnaround time is found to be 20105.65 seconds and all the jobs are completed at the 64420.25<sup>th</sup> second.

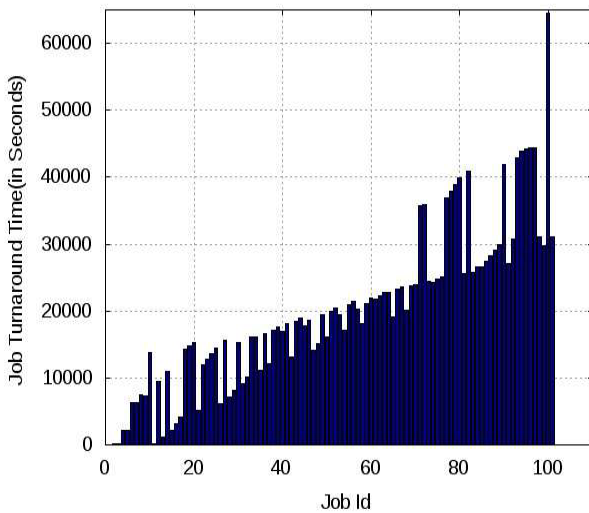


Figure 2. Jobs and turnaround time using Random Resource Selection.

**B. Equal Job Distribution**

In Equal Job Distribution we firstly calculate the total length of all the jobs and then distribute these length equally on every resource. The main notations which are used in the formula are as follows:

$L$  = Total length of all the jobs taken together.  $Proc\ power_i$  = Processing power of resource  $R_j$ .  $tProc\ power$  = Total processing power of all resources.  $Load_i$  = Load assigned on resource  $R_j$ .

The formula used to calculate the job distribution is given below:

$$Load_i = L * (Proc\ power_i / tProc\ power)$$

The turnaround time of each job is shown by the bar chart in Fig. 3. Experimental results show that the average turnaround time is 17968.55 seconds and the last result is outputted at 39000.22<sup>th</sup> second. Equal Job Distribution reduces the average turnaround time by 10.62% and it takes less time in comparison to the Random Resource Selection to give all the results.

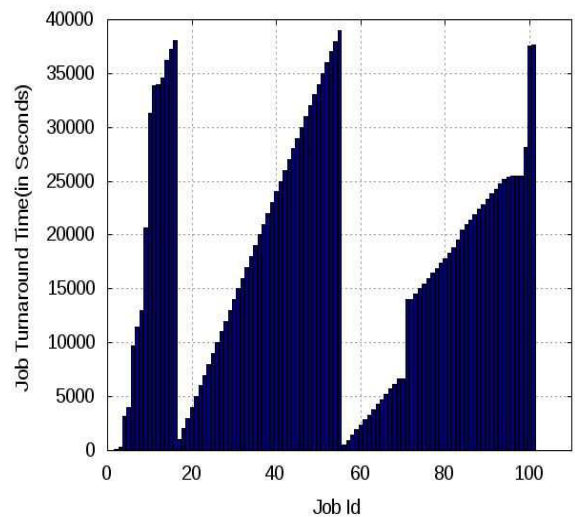


Figure 3. Jobs and turnaround time using Equal Job Distribution.

**C. Proposed Algorithm**

In Proposed Algorithm, the scheduler finds the resource information with the help of GIS and calculates the approximate completion time of this job on every resource. Using these values the scheduler chooses a resource which has the minimum of completion time and submits that job on this resource. The turnaround time of each job is shown in bar chart in Fig. 4. Through this algorithm the average turnaround time of these jobs is 17208.77 seconds and all the jobs are completed at 41840.88<sup>th</sup> second. The Proposed Algorithm further reduces the average turnaround time by 4.22% as compared with Equal Job Distribution. The completion time of all jobs takes some more time than Equal Job Distribution algorithm.

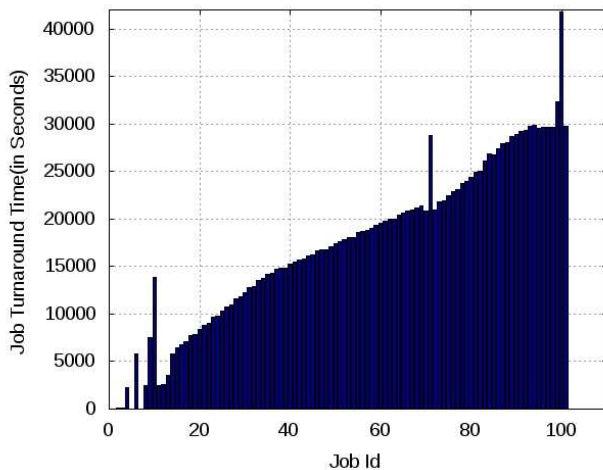


Figure 4. Jobs and turnaround time using Proposed Algorithm.

### V. CONCLUSION AND FUTURE WORK

The proposed scheduling algorithm reduces the average turnaround time of all submitted jobs. The considered environment executed the jobs on different resources which are geographically distributed. It is observed that the Proposed Algorithm reduces the average turnaround by 4.22% with Equal Job Distribution (as shown in Table II). The algorithm uses meta-scheduler where resource failure is not considered.

TABLE II. ALGORITHMS WITH THEIR AVERAGE TURNAROUND TIME AND COMPLETION TIME.

Algorithms	Average Turnaround Time (In Seconds)	Completion Time (In Seconds)
Random Resource Selection	20105.65	64420.25
Equal Job Distribution	17968.55	39000.22
Proposed Algorithm	17208.77	41840.88

### VI. REFERENCES

[1]. Alhusaini Ammar H., Prasanna Viktor K., Raghavendra C.S.(1999), "Unified Resource Scheduling Framework for Heterogeneous Computing Environments", in Proceedings of the Eighth Heterogeneous Computing Workshop, San Juan, Puerto Rico, (pp. 156-165).

[2]. Muthuvelu N., Liu J., Soe N. L., Venugopa S.R I, Sulistio A. and Buyya R.(2005), "A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids", Proceedings of the 3rd Australasian

Workshop on Grid Computing and e-Research (AusGrid 2005), Newcastle, Australia, (41-48).

[3]. Foster I., Kesselman C (1999), "The Grid: Blueprint for a new computing infrastructure", Morgan Kaufmann Publishers, San Francisco, USA.

[4]. Buyya R., Abramson D., Giddy J. (2000), "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computation Grid", International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China. IEEE Computer Society Press, USA.

[5]. Liu Cong, Baskiyar Sanjeev and Li Shuang (2007), "A General Distributed Scalable Peer to Peer for Mixed Tasks in Grids", High Performance Computing "HiPC 2007, ISBN: 978-3-540-77219-4, (320-330).

[6]. Buyya Rajkumar, Murshed Manzur (2002), "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", Technical Report, Monash University, Nov. 2001. To appear in the Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May 2002, (pp. 1-32).

[7]. Hamscher Volker, Schewiegelshohn Uwe, Streit Achim, Yahyapour Ramin (2000), "Evaluation of Job-Scheduling Strategies for Grid Computing", in 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000), Berlin, Lecture Notes in Computer Science (LNCS), Springer, Berlin, Heidelberg, New York, (pp. 191-202).

[8]. Cormen TH, Leiserson CE, Rivest RL (2001), "Introduction to algorithms 2nd edition", MIT and McGraw-Hill Book Company, Boston Massachusetts, (cp. 16, 370-403).

[9]. I. Foster, C. Kesselman, and S. Tuecke (2001). "The Anatomy of the Grid: Enabling Scalable Virtual Organizations.", International Journal of High Performance Computing Application, (15(3):200).

[10]. Rani Seema, Ritika (2014). "Grid Scheduling Algorithms: A Comparative Analysis", International Journal of Computer Applications. 104, 12 DOI = 10.5120/18255-9351, (26-29).

[11]. Sulistio A., Cibej U., Venugopal S., Robic B., and Buyya R (2008), "A toolkit for modelling and simulating data Grids: an extension to GridSim". Concurrency and Computation: Practice & Experience, (1591-1609).

[12]. Mehran Garmehi, Morteza Analoui, Mukaddim Pathan, Rajkumar Buyya (2015), "An economic mechanism for request routing and resource allocation in hybrid CDN P2P networks",



- International Journal for Network Management  
Int. J. Network Mgmt 2015, (375– 393).
- [13]. Huedo E., Montero R. S., and Liorente I. M (2005), “The GridWay framework for adaptive scheduling and execution on grids scalable computing: practice and experience” , (6(3)).
- [14]. Gokuldev S., Mathew Lija Anna (2013), “Task Graph Based Scheduling And Resource Monitoring In Grid Computing”, (vol. 2 Issue 5).
- [15]. Chtepen M. (2005), "Dynamic scheduling in grids system," Sixth Firw PhD Symposium, Faculty of Engineering, Ghent University.
- [16]. Carretero Javier, Xhafa Fatos, “Genetic algorithm based schedulers for grid computing”, ICIC international (volume 3, number 6)

# IJEAST

INTERNATIONAL JOURNAL  
OF ENGINEERING APPLIED SCIENCE  
AND TECHNOLOGY

## ABOUT IJEAST

International Journal of Engineering Applied Science and Technology (IJEAST) is a peer-reviewed, open access journal that publishes high-quality research papers in the field of Engineering, Applied Science and Technology.

IJEAST aims to provide a platform for researchers, academicians, and professionals to share their innovative ideas, research findings, and practical experiences with the global scientific community.

## FOCUS AREAS

- Engineering
- Applied Science
- Technology
- Innovation & Development
- Interdisciplinary Studies



### PEER REVIEWED

All submissions are rigorously peer reviewed to ensure quality.



### OPEN ACCESS

Free and unrestricted access to research for all.



### GLOBAL REACH

Connecting researchers and professionals worldwide.



### TIMELY PUBLICATION

We ensure a swift and efficient publication process.



For more information, visit our website

[www.ijeast.com](http://www.ijeast.com)



INTERNATIONAL JOURNAL  
OF ENGINEERING APPLIED SCIENCE  
AND TECHNOLOGY

✉ [editor@ijeast.com](mailto:editor@ijeast.com)

🌐 [www.ijeast.com](http://www.ijeast.com)

📍 India



2455-2143